WHAT IS CLAIMED:

1. A method of translating a plurality of basic blocks of program code, comprising:

decoding a plurality of basic blocks of program code;

generating an intermediate representation for each of said basic blocks of program code;

performing a partial dead code elimination optimization on said intermediate representation to generate an optimized intermediate representation;

generating target code from said optimized intermediate representation.

2. The method of claim 1, wherein the partial dead code elimination optimization is performed on basic blocks ending in non-computed branches or computed jumps.

3. The method of claim 2, wherein the partial dead code elimination optimization comprises:

identifying partially dead register definitions within a basic block;

applying a recursive marking algorithm to mark the child nodes of identified partially dead register definitions to produce a set of partially dead nodes; and

performing code motion optimization algorithm to generate an optimized intermediate representation providing an optimized order for generating target code.

4. The method of claim 3, wherein the partially dead register definition identifying step comprises:

for a register definition in a basic block, performing liveness analysis of said register definition in successor basic blocks containing said non-computed branches or computed jumps; and

identifying said register definition as being partially dead if said register definition is dead in at least one successor basic block and live in at least one other successor basic block.

5. The method of claim 4, further comprising forming a set of identified partially dead register definitions.

6. The method of claim 5, further comprising applying a recursive marking algorithm to identify partially dead child nodes in the intermediate representation of identified partially dead register definitions.

7. The method of claim 6, wherein said recursive marking algorithm identifies a node in the intermediate representation as a partially dead child node by ensuring that the node is not referenced by either a live node or a live register definition.

8. The method of claim 6, wherein said recursive marking algorithm identifies partially dead child nodes as those nodes which are only referenced in the intermediate representation by other partially dead nodes or partially dead register definitions.

9. The method of claim 8, wherein said recursive marking algorithm includes the steps of:

determining a dead count for a child node, wherein the dead count is the number of partially dead nodes referencing the child node in the intermediate representation;

determining a reference count for the child node, wherein the reference count is the number of references to the child node in the intermediate representation; and

identifying a child node as a partially dead when its dead count equals its reference count.

10. The method of claim 6, wherein said recursive marking algorithm further recursively identifies whether the child nodes of identified partially dead child nodes are also partially dead.

11. The method of claim 3, the code motion optimization algorithm comprises:

for each identified partially dead register definition,

determining a path of nodes in the intermediate representation for said partially dead register definition which are live,

discarding the nodes in the intermediate representation for said partially dead register definition which are dead, and

determining partially live paths of nodes in the intermediate representation for said partially dead register definition and moving corresponding nodes into said partially live paths, wherein the nodes in the partially live paths are partially

dead nodes, further wherein a partially live path of nodes exists for each respective branch or jump.

12. The method of claim 11, wherein each node in the intermediate representation includes an associated variable which identifies with which partially live path of nodes it is associated.

13. The method of claim 11, wherein said target code generating step comprises:

initially generating target code for all fully live nodes of a partially dead register definition; and

next generating target code for said partially live paths of nodes in the intermediate representation for said partially dead register definition.

14. The method of claim 11, wherein the code motion optimization algorithm further prevents consecutive load and store operations in the intermediate representation from being moved into one of the partially live paths.

15. The method of claim 3, further comprising the step of performing a load-store aliasing optimization.

16. A computer-readable storage medium having software resident thereon in the form of computer-readable code executable by a computer to perform the following steps to translate a plurality of basic blocks of program code:

decoding a plurality of basic blocks of program code;

generating an intermediate representation for each of said basic blocks of program code;

performing a partial dead code elimination optimization on said intermediate representation to generate an optimized intermediate representation; and

generating target code from said optimized intermediate representation.

17. The computer-readable storage medium of claim 16, wherein the partial dead code elimination optimization is performed on basic blocks ending in non-computed branches or computed jumps.

18. The computer-readable storage medium of claim 17, wherein the partial dead code elimination optimization comprises:

identifying partially dead register definitions within a basic block;

applying a recursive marking algorithm to mark the child nodes of identified partially dead register definitions to produce a set of partially dead nodes; and

performing code motion optimization algorithm to generate an optimized intermediate representation providing an optimized order for generating target code.

19. The computer-readable storage medium of claim 18, wherein the partially dead register definition identifying step comprises:

for a register definition in a basic block, performing liveness analysis of said register definition in successor basic blocks containing said non-computed branches or computed jumps; and

identifying said register definition as being partially dead if said register definition is dead in at least one successor basic block and live in at least one other successor basic block.

20. The computer-readable storage medium of claim 19, said computer-readable code further executable for forming a set of identified partially dead register definitions.

21. The computer-readable storage medium of claim 20, said computer-readable code further executable for applying a recursive marking algorithm to identify partially dead child nodes in the intermediate representation of identified partially dead register definitions.

22. The computer-readable storage medium of claim 21, wherein said recursive marking algorithm identifies a node in the intermediate representation as a partially dead child node by ensuring that the node is not referenced by either a live node or a live register definition.

23. The computer-readable storage medium of claim 22, wherein said recursive marking algorithm identifies partially dead child nodes as those nodes which are only referenced in the intermediate representation by other partially dead nodes or partially dead register definitions.

24. The computer-readable storage medium of claim 23, wherein said recursive marking algorithm includes the steps of:

determining a dead count for a child node, wherein the dead count is the number of partially dead nodes referencing the child node in the intermediate representation;

determining a reference count for the child node, wherein the reference count is the number of references to the child node in the intermediate representation; and

identifying a child node as a partially dead when its dead count equals its reference count.

25. The computer-readable storage medium of claim 21, wherein said recursive marking algorithm further recursively identifies whether the child nodes of identified partially dead child nodes are also partially dead.

26. The computer-readable storage medium of claim 18, the code motion optimization algorithm comprises:

for each identified partially dead register definition,

determining a path of nodes in the intermediate representation for said partially dead register definition which are live,

discarding the nodes in the intermediate representation for said partially dead register definition which are dead, and

determining partially live paths of nodes in the intermediate representation for said partially dead register definition and moving corresponding nodes into said partially live paths, wherein the nodes in the partially live paths are partially dead nodes, further wherein a partially live path of nodes exists for each respective branch or jump.

27. The computer-readable storage medium of claim 26, wherein each node in the intermediate representation includes an associated variable which identifies with which partially live path of nodes it is associated.

28. The computer-readable storage medium of claim 26, wherein said target code generating step comprises:

initially generating target code for all fully live nodes of a partially dead register definition; and

next generating target code for said partially live paths of nodes in the intermediate representation for said partially dead register definition.

29. The computer-readable storage medium of claim 26, wherein the code motion optimization algorithm further prevents consecutive load and store operations in the intermediate representation from being moved into one of the partially live paths.

30. The computer-readable storage medium of claim 18, further comprising the step of performing a load-store aliasing optimization.

31. In combination:

a target processor; and

translator code for translating a plurality of basic blocks of program code, said translator code comprising code executable by said target processor for performing the following steps:

decoding a plurality of basic blocks of program code;

generating an intermediate representation for each of said basic blocks of program code;

performing a partial dead code elimination optimization on said intermediate representation to generate an optimized intermediate representation; and

generating target code from said optimized intermediate representation.

32. The combination of claim 31, wherein the partial dead code elimination optimization is performed on basic blocks ending in non-computed branches or computed jumps.

33. The combination of claim 32, wherein the partial dead code elimination optimization comprises:

identifying partially dead register definitions within a basic block;

applying a recursive marking algorithm to mark the child nodes of identified partially dead register definitions to produce a set of partially dead nodes; and

performing code motion optimization algorithm to generate an optimized intermediate representation providing an optimized order for generating target code.

34. The combination of claim 33, wherein the partially dead register definition identifying step comprises:

for a register definition in a basic block, performing liveness analysis of said register definition in successor basic blocks containing said non-computed branches or computed jumps; and

identifying said register definition as being partially dead if said register definition is dead in at least one successor basic block and live in at least one other successor basic block.

35. The combination of claim 34, said translator code further comprising code executable by said target processor for forming a set of identified partially dead register definitions.

36. The combination of claim 35, said translator code further comprising code executable by said target processor for applying a recursive marking algorithm to identify partially dead child nodes in the intermediate representation of identified partially dead register definitions.

37. The combination of claim 36, wherein said recursive marking algorithm identifies a node in the intermediate representation as a partially dead child node by ensuring that the node is not referenced by either a live node or a live register definition.

38. The combination of claim 36, wherein said recursive marking algorithm identifies partially dead child nodes as those nodes which are only referenced in the intermediate representation by other partially dead nodes or partially dead register definitions.

39. The combination of claim 38, wherein said recursive marking algorithm includes the steps of:

determining a dead count for a child node, wherein the dead count is the number of partially dead nodes referencing the child node in the intermediate representation;

determining a reference count for the child node, wherein the reference count is the number of references to the child node in the intermediate representation; and

identifying a child node as a partially dead when its dead count equals its reference count.

40. The combination of claim 3, wherein said recursive marking algorithm further recursively identifies whether the child nodes of identified partially dead child nodes are also partially dead.

41. The combination of claim 33, the code motion optimization algorithm comprises:

for each identified partially dead register definition,

determining a path of nodes in the intermediate representation for said partially dead register definition which are live,

discarding the nodes in the intermediate representation for said partially dead register definition which are dead, and

determining partially live paths of nodes in the intermediate representation for said partially dead register definition and moving corresponding nodes into said partially live paths, wherein the nodes in the partially live paths are partially dead nodes, further wherein a partially live path of nodes exists for each respective branch or jump.

42. The combination of claim 41, wherein each node in the intermediate representation includes an associated variable which identifies with which partially live path of nodes it is associated.

43. The combination of claim 41, wherein said target code generating step comprises:

initially generating target code for all fully live nodes of a partially dead register definition; and

next generating target code for said partially live paths of nodes in the intermediate representation for said partially dead register definition.

44. The combination of claim 41, wherein the code motion optimization algorithm further prevents consecutive load and store operations in the intermediate representation from being moved into one of the partially live paths.

45. The combination of claim 33, said translator code further comprising code executable by said target processor for performing a load-store aliasing optimization.